高速数据包处理硬件加速技术

罗腊咏 谢应科 谢高岗

摘要：链路带宽的剧增给高速网络数据包处理带来了极大的挑战。传统的纯软件网络数据包处理在性能上已不能满足需要。当前网络处理器、多核芯片等针对高性能网络数据包处理提供了硬件加速技术，对多数网络应用提供了高性能实现方法。 在对数据处理时延、吞吐量、丢包率等性能指标有更高要求的应用场合，还需要专用的加速硬件。本文针对基于深度报文检测（DPI）的高性能流量分析和控制应用需求，介绍基于现场可编程逻辑门阵列（FPGA）的通用高速网络数据包处理硬件加速架构。该架构对数据采集通路进行硬件加速，实现了高速链路数据报文的线速采集，通过专用硬件进行数据包转发和流量控制，针对后端多核服务器的并行处理进行优化，实现了控制和分析平面的高性能处理。本文介绍该架构在流量采集、高精度时钟同步、高速包分类和流量控制等方面的硬件加速方法。测试结果表明，这些加速方法充分卸载了服务器的处理负荷，能有效地提高应用系统的性能。

关键词：数据采集 负载均衡 时钟同步 包分类 流量控制

1 引言

随着网络链路带宽的剧增和网络应用的多样化发展，网络数据包处理对计算资源和存储资源都提出了更高的要求。传统服务器的性能增长速度远远低于网络带宽的增长速度[1]，服务器的性能已经难以满足目前高速网络数据包处理的需要，因此针对高带宽和复杂网络应用的网络加速方法一直是学术界和工业界的研究热点。

以入侵检测、流量监控等应用为例，典型的数据包处理流程包括流量采集、预处理（时间戳标记、包分类、过滤等）、协议分析、流量控制等环节，如图 1 所示。

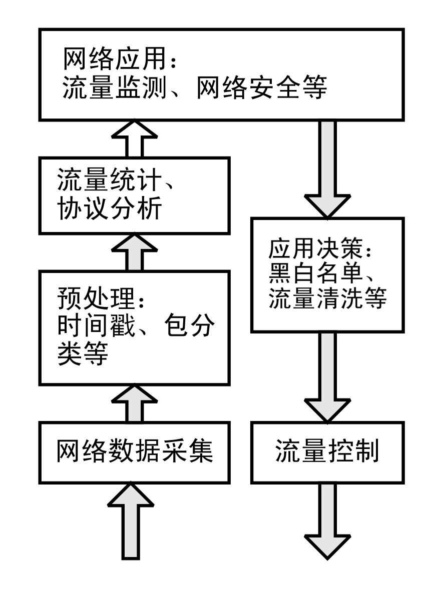


图1. 典型应用的数据包处理流程

图中，网络数据采集由普通网卡完成，数据包预处理（时间戳、包分类等）、流量统计与协议分析、应用决策、流量控制等复杂的数据包处理环节在后端服务器上实现。在系统实现上，要充分考虑 I/O 性能、主存性能以及计算性能。在高速网络链路环境下要使图 1 中所示的各个处理环节达到性能的均衡面临以下诸多挑战[2]：

1. **I/O 性能问题** 网络链路流量是所有网络分析和网络研究的基础。在高速网络环境下，对网络流量处理系统而言，网络数据采集的性能直接影响到网络流量处理系统的性能。在早期，由于网络速度较慢，应用较为单一，使用普通网卡与数据采集软件相配合，就能够实现网络流量的采集和处理。典型代表是 Libpcap[3]，BPF[4]，NPF[5] 等。然而，随着网络传输和光通讯技术的不断发展，骨干网链路速率已经达到了 10 Gbps。由于在 I/O 和数据传输通路上的不足，传统的网络数据采集方法已经失效，

更高的网络传输速率对网络流量处理系统的 I/O 性能和数据传输机制提出了巨大的挑战。

2. **主存性能问题** 业务流的多样性要求网络流量处理系统具备更高层次的处理能力。然而，随着处理层次的提高，实现高速网络环境下实时的网络流量处理也越发困难，表现为三个方面：

- **主存容量**：网络层流量处理系统对存储资源的需求与网络上的并发流数目成正比，而应用层流量处理系统对存储资源的需求与网络上的报文字节数目成正比。随着网络速率的不断提高，较高层次的网络流量处理对主存容量的需求巨大，成为一个不可忽视的问题。假设存储一条流的信息需要占用 200 字节的空间，在对一条有 500 万条流的 10 Gbps 网络链路进行处理时，网络层流量处理系统中仅用于流信息存储的主存空间需求就达到了 1 GB，如果考虑应用层流量处理，还需要存储每一个流中的所有载荷，此时，对主存容量的需求将会成倍增加。

- **访存带宽**：网络层流量处理是一种存储密集型操作，网络层流量处理的访存次数与数据报文数目成正比，而应用层流量处理的访存次数与网络链路中报文字节数目成正比。由于每一次内存访问都意味着所需的内存带宽加倍，在高速网络环境下，频繁的内存访问将会为主存带宽带来巨大压力。

- **访存延迟**：加州大学伯克利分校（UC Berkely）的研究小组认为，在计算机体系结构中，访存延迟带来的性能损失比访存带宽更为严重[6]。在网络流量处理系统中，这种问题同样存在。通常情况下，一次访存需要上百个 CPU 时钟周期。然而，在高速网络环境的极限情况下，每两个报文之间的时间间隔远小于每次访存所需要的时间。例如，在 OC-192 万兆链路上，40 字节长度的小报文的间隔时间仅为 34 纳秒，以 2 GHz 的 CPU 为例，与每两个报文间隔仅为 68 个处理器时钟周期相比，上百个周期的访存时间，显然已经无法满足高速网络流量的线速处理要求。

3. **CPU 性能问题** 如吉尔德定律[1]的描述，网络带宽以每 6 个月翻一番的速度持续增长。这一速度是摩尔定律预测的 CPU 主频增长速率的 3 倍。可以看到，网络链路速率与 CPU 处理能力之间的差距在逐步加大，用于处理每个数据报文的 CPU 时间越来越少。例如，传输 64 字节报文时，在千兆以太网，流量处理系统每秒需要处理约 149 万个数据包，而在 OC-192 万兆链路，每秒需处理的数据包数目达到了 1843 万，超过了前者的 10 倍，且要实现更高层次更复杂的网络流量处理。显然，已有 CPU 的处理能力已经远远不能满足线速处理要求。

为了解决纯软件实现的应用系统中上述环节的性能瓶颈问题，研究者开始考虑使用硬件加速技术，对数据包处理的各个环节进行硬件加速或者任务卸载，从而减轻 CPU 的处理负担，加速整个应用系统的性能。

本文首先简要介绍常见的硬件加速平台，包括网络处理器（NP）、现场可编程逻辑门阵列（FPGA）、专用集成电路（ASIC）和嵌入式多核处理器（Multicore Processor）等，并对这些平台的特点进行比较。针对对性能要求很高（如零丢包、μs 级延迟等）、同时要求具有一定灵活性的应用，基于现场可编程逻辑门阵列的硬件加速平台是一个很好的选择。本文介绍了一个基于现场可编程逻辑门阵列的通用高速网络数据包处理硬件加速架构及基于这种架构在流量采集、高精度时钟同步、高速包分类和流量控制等具体的硬件加速技术方面的一些研究工作。最后，对这些加速技术的性能表现进行评测。测试结果表明，基于现场可编程逻辑门阵列的硬件加速方法能有效地卸载服务器的处理负担，提高应用系统性能。

本文其余部分组织如下：第 2 节介绍常见的硬件加速技术，分析各种硬件平台的优缺点；第 3 节介绍基于现场可编程逻辑门阵列的硬件加速解决方案，针对在线流量分析和控制应用提出了一个通用的高速数据包处理硬件加速架构。第 4 节，介绍基于该架构的流量采集、高精度时间同步、高速包分类以及流量控制等方面的具体加速方法；第 5 节对相关加速技术进行性能评估；最后是对相关研究工作的简单总结。

2 高速数据包处理硬件加速技术

为了解决纯软件实现的应用系统在高速网络数据包处理中的性能瓶颈问题，学术界和工业界一直在研究高速网络数据包处理的硬件加速技术，出现了若干网络处理硬件加速平台。常见的硬件加速平台包括：网络处理器、嵌入式多核处理器、现场可编程逻辑门阵列和专用集成电路。

网络处理器是一类高性能可编程的专用指令集处理器。其内部通常包含若干个具有计算和存储能力的数据包处理引擎，多个处理引擎并行工作，可实现高速、复杂的网络数据处理。网络处理器的处理性能很高，具有代表性的产品包括英特尔公司的 IXP 系列[7]，IBM 公司的 Power NP 系列[8]等。然而，网络处理器采用微码编程，编程复杂，代码可复用和可移植性差，实际系统中使用得越来越少。

近几年来，迅猛发展的多核技术改变了业界的趋势，为网络算法和网络应用的并行处理提供了新的契机。一些公司推出的多核平台逐渐成为关注热点。典型的代表有 Cavium[9]公司的 OCTEON II 系列和 Tilera[10]公司的多核处理器 TILE64 和 TILE-Gx 系列。这类嵌入式多核处理器主要面向特定应用（如数字多媒体、网络数据包处理等）进行设计，因此，片内集成了专用的硬件加速引擎和专用的接口。针对网络数据包处理的需求，这类处理器通常集成了丰富的网络接口（如 SGMII，XAUI，Interlaken 等），以满足各种高速链路数据包采集的需求；集成了高速的存储接口（如 DDR3），以满足高速网络数据处理对存储带宽需求；集成了高速总线（如 PCIE 2.0），以解决与主机通信时的 I/O 瓶颈问题；集成了专用的硬件加速引擎，如针对网络安全应用的 Crypto Security 和针对深度报文检测的 HFA[1]引擎（用于模式匹配，Pattern Matching）等，面向特定应用的复杂处理环节进行专门的硬件优化；集成了大量处理核心，如 32 核、64 核、100 核，提供强大的并行处理能力，解决了传统 CPU 处理能力不足的问题；基于 C/C++的编程，大大增强了通用性、灵活性和可移植性。然而，多核架构引入的新问题，如核间通信延迟、数据共享开销等，在一些要求低转发时延、高吞吐量的应用中，可能成为性能瓶颈。另外，对于某些并行度不高的网络应用，多核并行处理的优势无法体现。

由于专用硬件逻辑能达到最高的性能，对于一些特定运算，如加解密、串匹配等，高性能的应用系统仍需要一些专用集成电路来实现。但专用集成电路的研发投入大、开发周期长、升级困难，目前只能适用于特定的应用。另外一种基于现场可编程逻辑门阵列的硬件加速技术获得了非常广泛的应用。现场可编程逻辑门阵列本身具备高速、并行和可编程特性，因此非常适合于高速网络数据包处理硬件加速。当前，现场可编程逻辑门阵列在容量和速度方面有了很大的发展，可以实现大规模的复杂逻辑，基于该平台的网络硬件加速方案得到了广泛研究和关注[11]。有代表性的研究和产品包括华盛顿大学 ARL 实验室的 FPX 平台[12]、斯坦福大学的 NetFPGA 项目[13]以及 Endace 公司开发的 DAG 系列监测卡[14]。

[1] Hyper Finite Automata，超级有限状态自动机

表 1 对上述硬件平台进行了归纳总结。

表 1 常见高速网络数据包处理硬件加速平台比较

硬件平台	主要优点	主要缺点	典型应用领域
网络处理器	集成了多个并行处理引擎，处理能力很强	微码编程困难，且代码的可复用性和可移植性差	传统的使用微码编程的网络处理器目前已很少使用，逐步被现场可编程门阵列和多核平台取代
现场可编程逻辑门阵列	高速、并行、可编程	灵活性有限，难以进行复杂的数据包处理	适用于对性能要求很高的特定应用领域
专用集成电路	高速，且可以在产品成型、量产阶段极大地降低成本	开发周期长、升级困难，不适用于研究平台	成熟的专用网络加速产品
嵌入式多核处理器	高并行度，集成了专用的网络数据包处理加速引擎，网络接口丰富，具有高速的总线接口	价格昂贵；并行编程、调试困难；多核架构带来核间通信延迟、数据共享开销等新问题	有很大并行度可以挖掘的网络数据包处理应用

综合来看，以上几种网络数据包处理技术还是针对提高 I/O 性能、主存性能以及计算性能来发展的。对于高速链路环境下的网络应用，都要采用分布式并行或者流水处理方法，通过硬件加速部件实现关键计算的卸载。本文针对在线流量分析和控制应用来讨论相关硬件加速技术。这类应用对数据包处理性能（如采集性能、转发延迟等）要求很高，控制层面的处理复杂，所以通常选用现场可编程逻辑门阵列作为硬件加速平台实现数据层面的线速处理，采用高性能服务器来做复杂的协议分析和策略控制。在硬件平台设计方面，核心是解决 I/O 性能、主存性能以及计算性能的均衡问题，优化整个处理流程，充分发挥系统的计算能力，实现整个系统性能的最优化。

3 高速数据包处理硬件加速架构

为了对传统的纯软件的数据包处理流程进行加速，本文在架构设计时主要遵循以下设计原则：

1. 保证线速采集：使用基于现场可编程逻辑门阵列的加速卡替代普通网卡，优化采集数据通路，实现高速流量的线速采集；
2. 充分发挥多核并行的优势：针对后端多核服务器并行处理进行硬件优化，在加速卡中利用多通道高速 DMA（直接内存访问）引擎和负载均衡技术，将流量均分给后端多个核处理，从硬件上直接支持数据并行化，充分发挥后端多核处理器的并行处理能力；
3. 充分卸载软件处理负担：利用现场可编程逻辑门阵列固有的并行和高速特性，将软件处理耗时，而硬件容易高速处理的功能模块在加速卡中实现，充分卸载后端服务器的处理负担。如硬件实现高精度时间戳、高速包分类、和高精度流量控制单元等。

基于上述考虑，并针对在线流量分析和控制等典型应用，我们设计了如图 2 所示的高速

数据包处理硬件加速架构。架构由控制平面和数据平面组成：数据平面完成数据包的各种处理，包括协议预处理、应用加速、数据采集和转发等；控制平面实现对系统的管理和配置，并向用户开放系统控制接口。

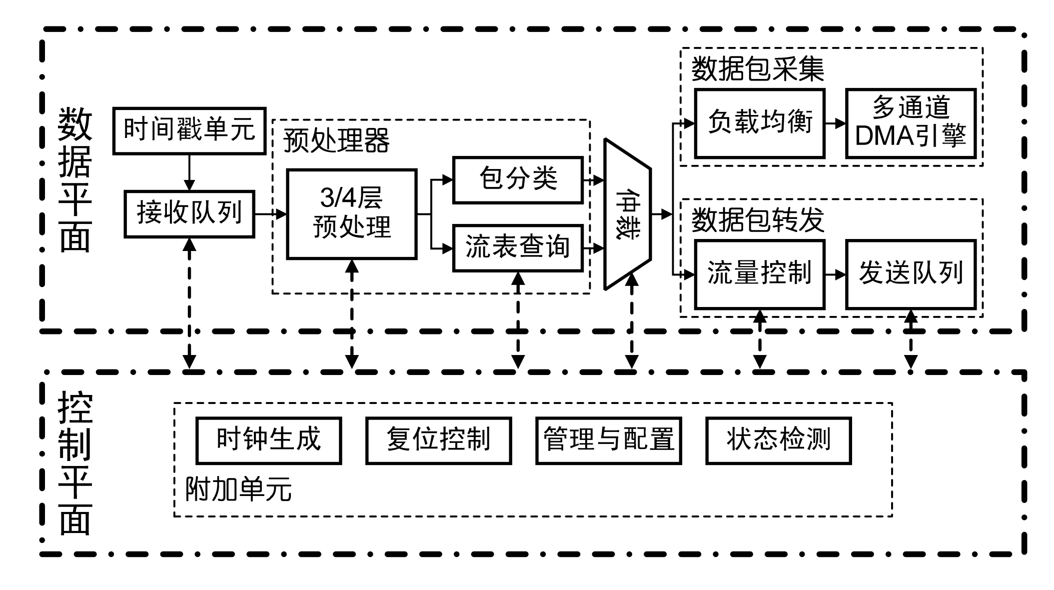


图2. 高速数据包处理硬件加速架构

数据平面的结构设计充分使用了流水和并行技术，如图 2 所示。每个数据包在进入接收队列时，立即被打上硬件时间戳。预处理部分首先对数据包的 3、4 层协议进行解析和预处理，然后根据不同的应用需求，通过可编程接口配置，有选择地使数据包进入包分类单元、流表查询单元等一组功能单元中的一个或同时进入若干个处理单元，进行后续处理。各个处理单元并行执行，处理结果通过仲裁模块进行仲裁。仲裁模块还要负责对数据包进行封装，将前端处理单元的处理结果和整个数据包封装在自定义格式的记录中，供后续处理使用。在流水级的后端，数据包被复制成两路：一路送入采集部分，通过负载均衡和多通道高速 DMA（直接内存访问）引擎传输至后端服务器进一步处理；另一路送入转发引擎，提供快速的数据包转发路径，根据具体的应用需求，可以选择对流量进行清洗和控制。

在系统设计中，要实现的关键技术包括：

- 在多核多处理器的处理环境中，通过硬件把网络数据流分发到相应的处理线程上，即硬件实现高性能的数据包分类——高性能的数据包分类算法及其实现是需要重点解决的问题。
- 实现高性能的数据采集，为了保证数据的不丢包，需要研究多缓冲队列、DMA（直接内存访问）技术、内存的循环缓冲技术、负载均衡技术等，从物理通道和处理机制上保证数据的高性能采集。
- 精确的流量控制技术，对于在线应用的系统来说，在硬件上实现流量控制可以有效降低时延，实现可扩展的高精度流量控制单元是保证系统可用性的重要工作。
- 相关硬件加速单元的实现，例如，网络分析需要精确分析数据包的时间序列关系，对时间戳有着很高精度的要求，在 OC192 链路中，数据包的最小时间间隔仅为 30.5ns[15]，要求时间戳的分辨率在 ns 级。通过在硬件上产生时间戳，既可以保证高精度，也可以有效降低软件的运算量。

4 高速数据包处理的硬件加速方法

基于上节提出的加速架构，我们课题组在数据采集、时钟同步技术、包分类技术、流量控制技术等方面进行了一系列研究工作，以下分别从这些方面进行介绍。

4.1 并行优化的线速数据采集结构

网络流量采集是对网络流量进行全面统计分析的基础。传统的流量采集通常基于软件实现，使用普通网卡与通用 CPU 相组合，完成网络流量的采集与分析[16-17]。这种方式具有简单易用、可移植性好等特点。然而，粗略估计[18]，每处理 1 比特的网络数据，需要消耗 1Hz 的 CPU 处理能力。随着链路带宽的提升，传统的采集方式在中断、内存拷贝等环节存在瓶颈，无法保证在高速链路数据采集中不丢包。随着现场可编程逻辑门阵列在容量、速率上的不断升级，基于该平台的硬件加速方案得到了广泛研究和关注[11]。有代表性的研究和产品包括华盛顿大学 ARL 实验室的 FPX 平台[12]、Endace 公司开发的 DAG 系列监测卡[19]。Endace 公司基于现场可编程逻辑门阵列实现的高速流量采集卡（DAG 卡），可以实现 1G、2.5G 及 10G 链路环境下的线速采集，在数据捕获过程中几乎不占用任何 CPU 资源，将所有的处理能力提供给应用程序。

为了进一步提高数据采集性能，并针对后端多核服务器的并行处理进行优化，我们基于现场可编程逻辑门阵列提出了一个并行优化的线速数据采集结构，如图 3 所示。

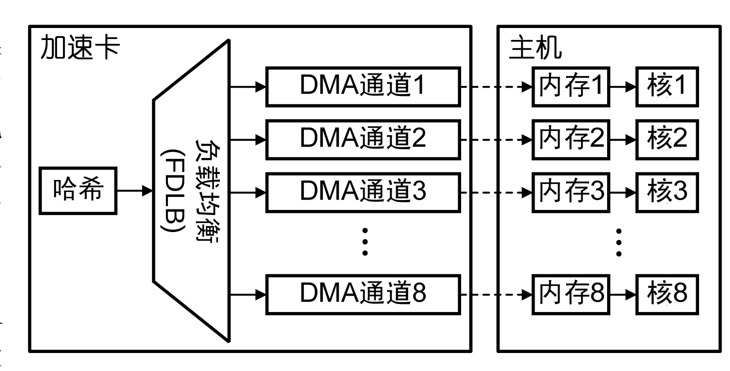


图3. 并行优化的采集结构

传统的采集结构中，DMA（直接内存访问）引擎只需要设计一个传输通道。为了卸载软件多线程共享数据的加锁开销，我们基于现场可编程门阵列设计了一个 8 通道独立的高速 PCIE DMA（直接内存访问）引擎。使用负载均衡单元在硬件中实现数据分发，然后将数据包通过 8 个 DMA（直接内存访问）通道分别高速传输到主机的不同内存缓冲区。因此，数据包进入后端服务器时，已经实现了数据级的并行分解，后端各处理线程数据完全独立，可以实现无锁编程，避免了线程互斥开销。同时，后端主机可以配合使用环形缓冲区技术和零拷贝技术[20]，避免数据拷贝的开销。在 DMA（直接内存访问）引擎设计时，为了更适应高速链路的数据采集需求，我们设计的 DMA（直接内存访问）引擎支持轮询机制控制，从而避免中断开销。

后端多核并行处理单元的处理能力可能不同，如果不考虑后端并行处理单元的能力差异，将流量均分处理，可能导致处理能力强的单元经常空闲，而处理能力差的单元无法及时处理所有数据包。因此，我们提出了一个基于反馈的负载均衡算法（FDLB，Feedback based Dynamic Load Balance）[21-22]，该算法可根据处理单元负载能力动态调整负载分配，且能尽量保持会话完整性。当输入流量大于后端处理能力时，负载均衡单元中引入随机早期丢弃算法（RED）[23]尽早地丢弃数据包，以避免拥塞的发生。

4.2 高精度时钟同步技术[21]

网络应用，如流量工程、服务质量等都需要时间信息[24-27]。时钟信息的精度将直接影响

应用系统计算结果的准确度。高精度时钟同步包括以下两个问题[21]：

1. **单系统时钟分辨率：** 系统时钟分辨率应能够区分出链路上相邻到达的数据包；
2. **多系统时钟同步：** 分布式网络应用系统的多个节点时钟要保持精确同步。

在纯软件的时间同步方法中，网络时间协议 NTP（Network Time Protocol）应用最为广泛[21]。节点利用 NTP 协议与网络中时间服务器同步，通过估算数据包在网络上往返延迟，计算节点时钟与时间服务器时钟偏差，从而完成时钟同步。NTP 可以达到毫秒级同步精度[28]。文献[29]基于实时 Linux 操作系统，利用 NTP 协议获得了误差在 1μs 之内的稳定时钟。软件时钟同步方法实现简单，但是精度不高，性能不稳定。霍拉尔(Horauer)等人提出了一种基于以太网的时钟同步技术 SynUTC[30]，利用硬件产生时间戳，其同步误差为 100ns。文献[31]设计了专用数据包捕获 DAG 卡，它利用硬件给所有捕获的数据包加时间戳，所有卡与 GPS 时钟同步，时间戳偏差在 100ns 之内。由于要用到 GPS，系统价格成本过高。

为了满足众多网络应用对时钟同步的需求，我们课题组提出了一种软硬件结合提高系统时钟同步精度的方法[21]。该方法利用硬件产生时间戳，消除了各种软件延迟的影响。同时设计了基于预测的时钟同步算法 PCS，利用该算法完成各节点的时钟同步。设计的时钟同步子系统具有相当的灵活性：既可以使用 GPS 时钟作为时钟源完成各节点的同步；也可以在不使用 GPS 时钟源的情况下，利用串口通信实现多个节点的同步，且同步精度不低于使用 GPS 的情况。

硬件时间戳生成电路如图 4 所示[21]。

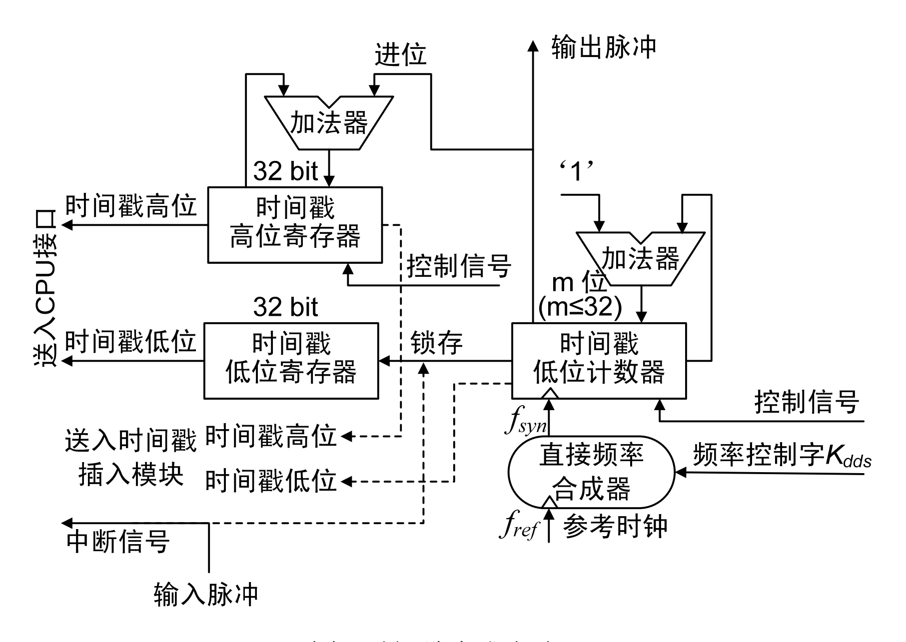


图4. 时间戳生成电路

时钟同步实际上就是调整一个系统的时钟频率使其与参考时钟频率一致的过程。在时间戳生成电路中，这部分功能由直接数字频率综合器（DDS, Direct Digital Synthesizer）完成。通过调整 DDS 频率控制字，可以获得各种频率的时钟。PCS 算法[21]的原理可以图 5 描述。

算法分成启动和同步两个阶段。在启动阶段，从节点完成时间戳秒分数部分的偏差纠正；在同步阶段，从节点基于上次结果微调本次 DDS 频率控制字，使得自身时钟与主节点同步。

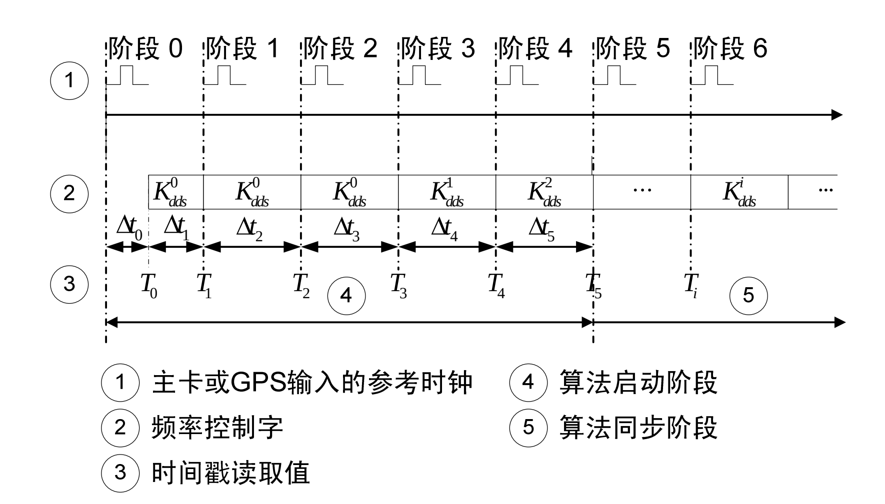


图5. PCS 算法示意图

4.3 通用包分类结构 RSTCAM[21]

包分类技术为在网络设备中把到达的数据包归类为不同的数据流提供了支持。它用不同的规则来标识各个数据流，每条规则根据对数据包头部各字段的分析指出该数据流中的数据包应当执行的相应操作。典型的包分类算法主要包括 3 类[21]：基于决策树的算法，基于分解-并行的算法和基于 TCAM[2]的算法。基于决策树的算法是对线性搜索算法的改进，它采用树结构代替传统的线性链表结构，首先根据分类规则集建立一棵决策树，分类时数据包沿着决策树查找获取分类结果。此类算法的典型代表有 Grid-of-Tries 算法[32]、FIS-tree 算法[33]、AQT 算法[34]、EGT-PC 算法[35]、HiCuts 算法[36]以及 HyperCuts 算法[37]等；基于分解-并行的算法核心思想是将分类问题并行化，通过分解将多维包分类问题转化成多个一维包分类问题（如 BV 算法[38]、ABV 算法[39]），或者将一个大空间的分类问题分割成多个子空间的分类问题（如 Tuple Space 算法[40]），降低算法复杂度，提高算法性能。基于 TCAM 的算法引入硬件解决方案，由于 TCAM 具有极快的分类速度，目前商用包分类器均采用基于 TCAM 的包分类。但是，由于 TCAM 中大量的并行结构导致很高的功耗，在存储密度、可扩展性方面也存在很多限制。因此此类算法的研究致力于寻求解决上述问题的办法。具有代表性的包括 ETCAM 算法[41]、EaseCAM 算法[42]、CoolCAMs 算法[43]等。

对于常见的基于五元组的包分类算法而言，由于五元组中 5 个域相互独立，可以先对每个域进行匹配，再将匹配结果进行合并得到最终匹配结果。文献[38]最早提出了这种想法，并利用 5 片 128Kb SRAM 实现了一个包分类器。该分类器支持 512 条规则，平均每秒可执行 100 万次查找。我们基于类似的思想，在 RSTCAM[21]的设计中，使用了 5 个 TCAM，每个 TCAM 存储五元组中一个域，将各域匹配结果进行“与”运算，得到最终分类结果。RSTCAM 结构如图 6 所示[21]。

RSTCAM 将原来五维查找转换成了 5 个一维查找，由此带来以下优点[21]：

1. 提高工作频率：TCAM 工作频率与其位宽有关，位宽越大，工作频率越低。通过将五维查找转换为 5 个一维查找，降低了每个 TCAM 的位宽，因此工作频率得到了较大提升。
2. 减少资源占用量：在分类规则集中通常有许多具有相同域的规则，通过合并相同域，每个 TCAM 的容量可以减少很多。

[2] Ternary Content Addressable Memory，三态内容可寻址存储器

3. 降低系统功耗：TCAM 功耗主要由其内部并行结构产生，采用 RSTCAM 结构降低了 TCAM 并行结构的位宽，从而降低了功耗；另外，由于 5 个 TCAM 相互独立，可以单独打开、关闭某个 TCAM，实现分类精度与功耗之间的权衡。

4. 易于实现范围匹配：分类规则中以范围表示的域通常需要将其表示转换成前缀形式。最坏情况下一条 w 比特范围表示需要转换成 $2(w-1)$ 条前缀。设规则 $\boldsymbol{R_i}$ 包含 $\boldsymbol{d}$ 个域，每个域均为范围表示，每个域的位宽分别为 $\boldsymbol{w_1, w_2, \cdots, w_d}$，则最坏情况下规则 $\boldsymbol{R_i}$ 需要转换成 $2(\boldsymbol{w_1}-1)\times 2(\boldsymbol{w_2}-1)\times\cdots\times 2(\boldsymbol{w_d}-1)$ 条前缀规则。采用 RSTCAM 结构后，由于各域相互独立，最坏情况下仅需 $2(\boldsymbol{w_1}-1)+2(\boldsymbol{w_2}-1)+\cdots+2(\boldsymbol{w_d}-1)$ 条规则。

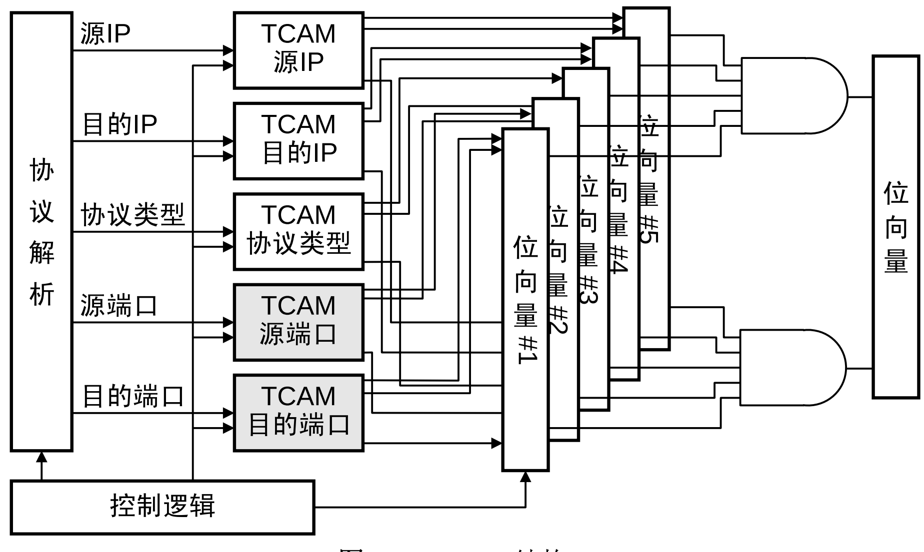


图6. RSTCAM 结构

4.4 高精度可扩展的流量控制技术

流量控制是网络监测、网络安全、服务质量保证等网络应用的关键技术。任丰原等人[44]按照历史的演进，将流量控制研究分为两个阶段：端到端的流量控制（主机）和中间节点上的流量控制（路由器）。早期的流量控制主要利用 TCP 中的滑动窗口机制在端系统上实现。后来很多研究人员继续对其做了深入的研究，不断改进算法中存在的缺陷，出现了 5 个主要版本的 TCP 流量控制算法：Tahoe[45]，Reno[46]，NewReno[47]，SACK[48]和 Vegas[49]。随着技术的发展，研究者开始意识到，仅仅在端系统上实施流量控制，很难满足诸如服务质量等要求。于是，大量的研究工作开始关注网络中间节点设备（路由器）上的流量控制。路由器上的流量控制研究主要集中在队列管理和队列调度。队列管理算法研究何时丢弃何种报文，以维持较小的队列长度；队列调度算法决定多个等待队列中的分组发送的次序，决定各队列的带宽分配。典型的队列管理算法包括 RED 算法[23]以及各种基于 RED 的改进算法，如 ARED(Adaptive RED)[50]，Balanced-RED[51]等。典型的队列调度相关研究工作包括：GPS（Generalized Processor Sharing）模型[52]，加权轮询队列 WRR（Weighted Round Robin）[53]和加权公平队列 WFQ（Weighted Fair Queue）[54]等。

由于路由器上的流量控制基于队列管理和队列调度实现，而队列的资源是有限的，很难满足业务流级带宽精细管理的需求。为了解决这个问题，本文提出了一种“虚通道”的概念。

虚通道由虚通道号（Virtual Channel ID）标识。每个数据包携带一个虚通道号，根据携带的虚通道号选择进入不同的虚通道。在每个虚通道的入口，设计一个控制开关，如图 7 所示。

在图 7 中，每个虚通道入口的控制开关本质上是一个数据包准入检查单元。该单元检查每个到达虚通道入口的数据包，决定是否允许其进入虚通道。如果不允许，则丢弃该数据包。该单元主要包含如下几个关键参数：

$\boldsymbol{T}$：基本的时间单元。本文将时间轴划分为若干个时间单元 $\boldsymbol{T}$，在每一个时间单元内实施流量控制。

$\boldsymbol{B}$：在时间单元 $\boldsymbol{T}$ 内，进入虚通道的数据报文的总长度计数值。在每个时间单元 $\boldsymbol{T}$ 的初始时刻，$\boldsymbol{B}=0$。

$\boldsymbol{W}$：数据包总长度阈值；每个时间单元 $\boldsymbol{T}$ 内，进入虚通道的数据包总字节数 $\boldsymbol{B}$ 不能超过 $\boldsymbol{W}$。

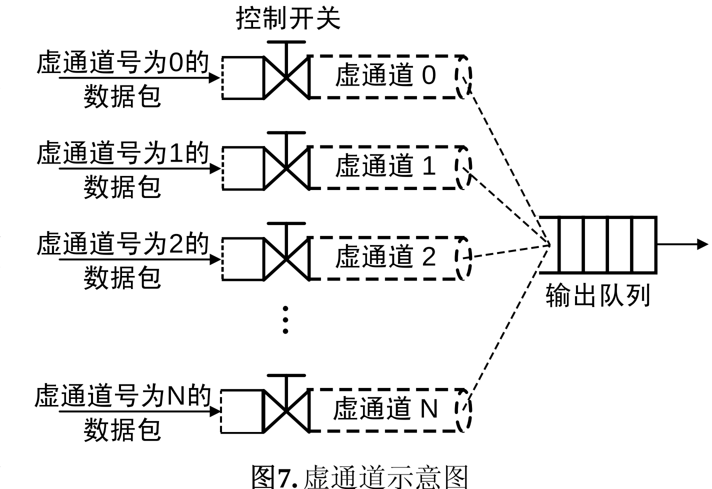


图7. 虚通道示意图

设时间单元 $\boldsymbol{T}$ 内进入的第 $\boldsymbol{i}$ 个数据包的长度为 $\boldsymbol{L_i}$，第 $\boldsymbol{i}$ 个数据包允许进入虚通道的条件是：

$$\boldsymbol{B}+\boldsymbol{L_i}\le\boldsymbol{W}$$

如果上述条件不满足，第 $\boldsymbol{i}$ 个数据包将被丢弃。如果第 $\boldsymbol{i}$ 个数据包成功进入虚通道，则修改 $\boldsymbol{B}=\boldsymbol{B}+\boldsymbol{L_i}$，否则 $\boldsymbol{B}$ 保持不变。

对于所有的时间单元 $\boldsymbol{T}$，如果保持 $\boldsymbol{W}$ 参数不变，那么，我们就可以使保证虚通道的带宽 $\boldsymbol{R}$ 满足：$\boldsymbol{R}\le\boldsymbol{W}/\boldsymbol{T}$。这就是流量限制的情形；

如果第 $\boldsymbol{m}$ $(\boldsymbol{m}>1)$ 个时间单元的阈值 $\boldsymbol{W_m}$，根据第 $(\boldsymbol{m}-1)$ 个时间单元最后时刻的 $\boldsymbol{W_{m-1}}$ 和 $\boldsymbol{B_{m-1}}$ 进行如下调整：$\boldsymbol{W_m}=\boldsymbol{W}+(\boldsymbol{W_{m-1}}-\boldsymbol{B_{m-1}})$，那么，该虚通道的流量大小 $\boldsymbol{R}$ 就能满足：$\boldsymbol{R}\approx\boldsymbol{W}/\boldsymbol{T}$，这就是流量精确控制的情形。然而，如果一直按照上式修正，当实际流量小于控制值时，$\boldsymbol{W_m}$ 将无限制的变大，最后，当实际流量高于控制值时，流量会在很长一段时间内无法控制。因此，$\boldsymbol{W_m}$ 超过一定阈值时必须将其修正部分清零，使 $\boldsymbol{W_m}=\boldsymbol{W}$。

虚通道本身并不消耗任何实际的逻辑资源，它仅仅是多业务流同时进行控制时，对各个流的数据通路的一种抽象。真正消耗逻辑资源的是每个虚通道入口的控制开关。在现场可编程逻辑门阵列中，要实现 1024 个虚通道，需要 1024 个参数不同的控制开关。然而，无论当前链路中有多少并发流，数据包在微观上是串行传输并处理的。因此，实际上不需要设计 1024 个不同的控制开关，而是要实现存储 1024 个不同虚通道的控制参数（$\boldsymbol{T}$、$\boldsymbol{B}$、$\boldsymbol{W}$ 等）的一个控制单元。对进入控制单元的每一个数据包，根据其携带的虚通道号，读出该数据包对应虚通道的控制参数，对其进行准入检查。处理完一个数据包后，将修改后的参数 $\boldsymbol{B}$ 存储起来，并准备处理下一个数据包。

另外，时间单元参数 $\boldsymbol{T}$ 可以在各个虚通道之间全局共享，因此，要实现 1024 种不同的流量控制策略，需要 1024 个虚通道，消耗的逻辑资源包括：一个统一的控制单元，1024 个参数 $\boldsymbol{B}$、$\boldsymbol{W}$ 的存储，一个全局参数 $\boldsymbol{T}$ 的管理。因此，每增加一个控制通道，仅仅只需要增加两个参数的存储资源，该方法对流量控制通道数量有很好的可扩展性。

最后，还需要一个大的输出缓存队列，对所有的流进行统一整形。由于数据包的传输在微观上是串行的，多个虚通道的网络流量汇聚到输出队列时，不需要进行调度。

5 性能评测

我们基于赛灵思（Xilinx）公司的 V5 系列现场可编程逻辑门阵列实现了数据包处理硬件加速的多个原型系统，以满足不同的网络链路的需求，如 GE(Gigabit Ethernet)，10GE，10G POS（Packet Over Sonet）。利用思博伦（Spirent）公司的 TestCenter 网络测试仪对系统的加速性能进行了全面测试，主要评测包括如下几个方面：

1. 线速采集测试[2]

图 8 列出了各种长度数据包测试流量下，数据采集的包速率。除了最极端的情况（包长度为 64 字节），加速平台都能够实现万兆网络流量的线速采集。在数据包长度为 64 字节的极限情况下，存在少量的数据包丢失。为了减轻后端 CPU 的处理压力，传输的每个数据包都附加了 16 字节的额外信息，占用 PCI-Express 总线带宽，使得 PCI-Express 总线传输的实际带宽高于测试仪表的发送带宽。在四 CPU 同时进行数据采集时，根据测试结果（采集的最大包速率）计算可知，此时 PCI-Express 总线的有效载荷（payload）带宽已经达到了 11.2Gbps，接近理论极限带宽，从而引发丢包。

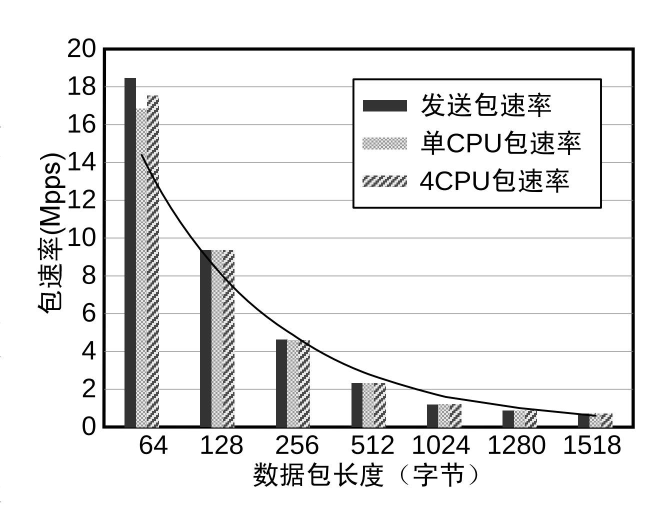
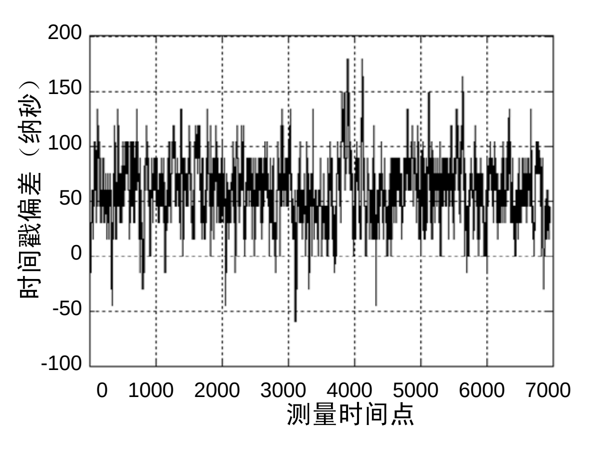


图8. 数据采集测试



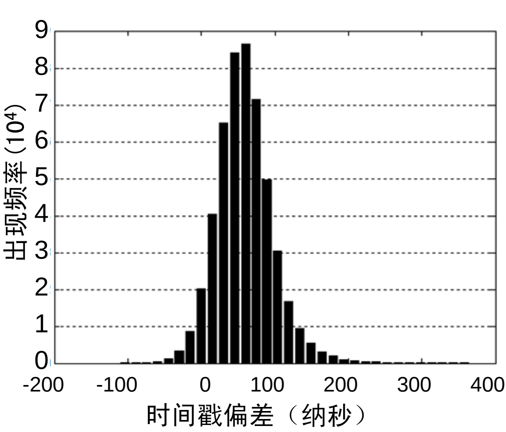


图9. 主、从节点同步的时间戳偏差及偏差分布

2. 双卡时钟同步精度测试[21]

将两块加速卡安装到不同服务器中，通过 RS422 互连两块加速卡。首先在两个服务器上运行网络时间协议 NTP，使其达到秒级同步，然后启动 PCS 同步算法，并且分别设置主、从工作模式，测得主、从节点时间戳偏差结果如图 9 所示。限于篇幅，图 9 左半部分中只描绘了 2 小时共 6956 个点的图形，其右半部分是对 12 小时测量结果的统计。从结果可见，PCS

算法通过调整频率控制字很好地完成了两块卡的同步，大部分时间戳偏差控制在 0～100ns 之内。该结果达到了基于 GPS 同步的精度。

3. 包分类功能测试[21]

为验证包分类功能，设置包分类器第 1 条规则[X, X, UDP, X, X]匹配所有 UDP[3]流量；第 10 条规则[192.168.0.X, X, TCP, X, X]匹配所有源 IP 地址为 192.168.0.X TCP 流量；第 50 条规则[192.168.1.X, 192.168.100.X, TCP, X, X]匹配所有源、目的 IP 地址为 192.168.1.X 和 192.168.100.X TCP 流量；第 150 条规则[192.168.1.X, 192.168.200.X, TCP, 1-255, X]匹配所有源、目的 IP 地址为 192.168.1.X 和 192.168.200.X 且源端口号 1-255 的 TCP 流量；设置最后一条规则匹配所有流量；其它规则全 0。

仪表生成以下 5 种流量：stream0 为 UDP 流量；stream1 为 TCP 流量且源 IP 为 192.168.0.X；stream2 为 TCP 流量且源、目的 IP 分别为 192.168.1.X 和 192.168.100.X；stream3 为 TCP 流量，源、目的 IP 分别为 192.168.1.X 和 192.168.200.X，源端口号 1-255；stream4 为 TCP 流量，源、目的 IP 分别为 192.168.1.X，192.168.200.X，且源端口号 1024-4095。所有测试流量均为小包 64 字节，每种流量占总带宽 20%，测试时逐步将匹配规则 1、10、50、150 的流量过滤，测得各 stream 带宽如图 10 所示。

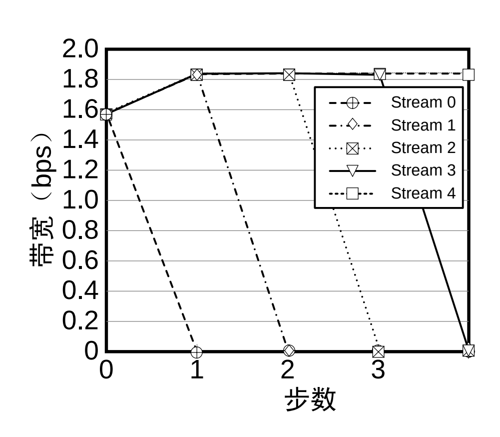


图10. 包分类功能测试

第 0 步时端口上各流量带宽相等（157446808bps），到第 1 步时将匹配规则 1 的流量过滤（通过程序修改与分类规则 1 相对应的查找表中的操作码，将其置为 0xf，过滤匹配的数据包），stream0 带宽变成 0，其它流量带宽变成 184325000bps（此时除过滤的包之外，不存在丢包），到第 2 步时将匹配规则 2 的流量过滤，stream1 带宽变成 0，到第 4 步时只剩下 stream4 流量。图 10 的测试结果说明了 RSTCAM 包分类引擎能在万兆速率下正确分类数据包。

4. 流量控制精度测试

使用 TestCenter 产生包长随机、速率为 100%的测试流量，通过我们加速卡的流量控制功能，将流量分别控制为 100Kbps，1Mbps，10Mbps，50Mbps，100Mbps 及 200Mbps，每次测试时间为 30 分钟，每秒钟采样一个速率值，流控精度测试结果如图 11 所示。

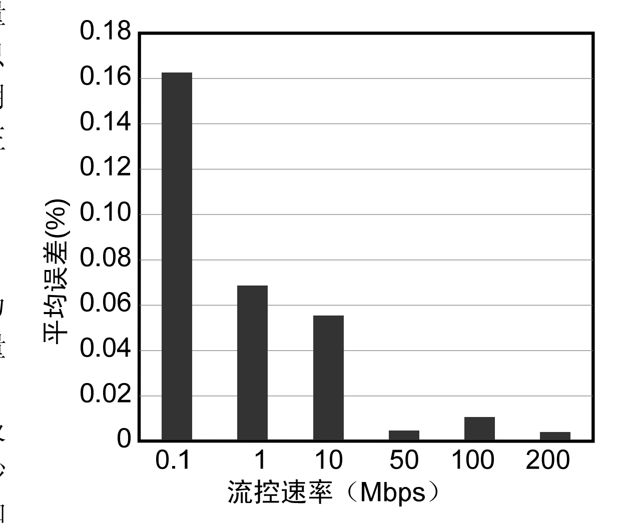


图11. 流量控制精度测试

测试结果表明，流量控制值越大，相对误差越小。小流量情况下误差大致在 0.1%左右，大流量情况下，平均误差都在 0.1%以下。

3 User Datagram Protocol，用户数据包协议

6 总结与展望

高速海量的网络数据和更加复杂的处理需求，给高速网络数据包处理带来了巨大的挑战。传统服务器的性能远远无法满足目前高速数据包处理的需求，高速数据包处理在很多环节存在性能瓶颈。为了全面提升高速网络数据包的处理性能，我们提出了一个通用的基于现场可编程门阵列的高速网络数据包处理加速架构。该架构有如下三个特点：（1）高速流量线速采集；（2）针对多核并行处理进行优化；（3）硬件充分卸载软件负担；基于该架构，我们课题组在流量采集技术、高精度时钟同步技术、高速包分类技术和流量控制技术等方面，提出了具体的硬件加速方法。测试结果表明，我们提出的硬件加速方法行之有效，能有效的提高应用系统的整体性能。

基于课题组已有的研究成果，可以在如下方面进一步深入研究：

1. 研究更高速链路（如 40Gbps、100Gbps）数据包处理的关键技术，以满足骨干网带宽不断升级的需求；
2. 研究应用层深度报文检测的硬件加速技术。深度报文检测是协议分析、网络安全等应用的关键技术，传统的软件算法很难满足高速链路深度报文检测的性能需求，因此，更高速率、更大规模规则集的深度报文检测的硬件加速技术是一个值得研究的方向。
3. 基于多核平台的高速数据包处理加速技术。现场可编程逻辑门阵列本身固有的并行、高速和可编程特性，使其非常适合对高速数据包处理进行硬件加速。然而，现场可编程逻辑门阵列本身的灵活性有限，通用性不强，很难满足复杂的网络处理需求。多核（Multi-Core）、众核（Many Core）技术的兴起，给高速数据包处理提供了新的解决方案。如何充分挖掘数据包处理的并行度、充分发挥多个 CPU 并行处理的优势，是一个值得深入研究的方向。

参考文献：

[1] G. Gilder, "Fiber Keeps Its Promise: Get ready. Bandwidth will triple each year for the next 25," *Forbes (Apr)*, 1997.

[2] 祝超, "高速网络流量处理系统优化方法研究," 博士学位论文, 中国科学院研究生院, 2010.

[3] V. Jacobson, *et al. Libpcap*. Available: http://www.tcpdump.org

[4] M. Steven and J. Van, "The BSD packet filter: a new architecture for user-level packet capture," presented at the Proceedings of the USENIX Winter 1993 Conference Proceedings on USENIX Winter 1993 Conference Proceedings, San Diego, California, 1993.

[5] F. Risso and L. Degioanni, "An Architecture for High Performance Network Analysis," presented at the Proceedings of the Sixth IEEE Symposium on Computers and Communications, 2001.

[6] K. Asanovic, *et al.*, "The Landscape of Parallel Computing Research: A View from Berkeley," Electrical Engineering and Computer Sciences, University of California at Berkeley2006.

[7] N. Shah, *et al.*, "NP-Click: A programming model for the Intel IXP1200," in *2nd Workshop on Network Processors (NP-2) at the 9th International Symposium on High Performance Computer Architecture (HPCA-9), Anaheim, CA*, ed, 2003.

[8] J. Allen Jr, *et al.*, "IBM PowerNP network processor: Hardware, software, and applications," *IBM Journal of Research and Development*, vol. 47, pp. 177-177, 2003.

[9] Cavium. *OCTEON Multi-Core Processor*. Available: http://www.caviumnetworks.com/

[10] Tilera. *Multicore Processors*. Available: http://www.tilera.com/

[11] G. Sun and Z. S. He, "A Real-Time Multi-Channel Signal Acquisition Card Based on PCI Express Interface," *Proceedings of the International Conference on Communication Software and Networks*, pp. 20-24 848, 2009.

[12] J. W. Lockwood, *et al.*, "Reprogrammable network packet processing on the field programmable port extender (FPX)," in *9th International Symposium on Field Programmable Gate Arrays(FPGA)*, Monterey, California, United States, 2001, pp. 87-93.

[13] J. Lockwood, *et al.*, "NetFPGA--an open platform for gigabit-rate network switching and routing," 2007, pp. 160-161.

[14] S. Donnelly and P. Limited, "Dag packet capture performance," *White Paper, August*, 2006.

[15] S. F. Donnelly, "High Precision Timing in Passive Measurements of Data Networks," Ph D, University of Waikato, 2002.

[16] V. Y. Hnatyshin and A. F. Lobo, "Undergraduate data communications and networking projects using opnet and wireshark software," presented at the Proceedings of the 39th SIGCSE technical symposium on Computer science education, Portland, OR, USA, 2008.

[17] L. Deri, "nProbe: an Open Source NetFlow Probe for Gigabit Networks," in *TERENA Networking Conference*, 2003.

[18] E. Yeh, *et al.*, "Introduction to TCP/IP offload engine (TOE)," *10 Gigabit Ethernet Allience Version*, vol. 1, 2002.

[19] Endace. (2010, *DAG Network Monitor Cards*. Available: http://www.endace.com/

[20] 赵自力, "万兆网络流量处理系统性能优化技术研究," 硕士学位论文, 中国科学院研究生院, 2009.

[21] 王建东, "万兆主干网数据采集与预处理关键技术研究," 博士学位论文, 中国科学院研究生院, 2009.

[22] 王建东，祝超，谢应科，韩承德，赵自力, "基于 FPGA 的万兆流量并行实时处理系统研究," *计算机研究与发展*, vol. 46, pp. 177-185, 2009.

[23] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," *IEEE/ACM Transactions on networking*, vol. 1, pp. 397-413, 1993.

[24] B. B. K. Salamatian, and T. Bugnazet, "Cross traffic estimation by loss process analysis," presented at the In Proceedings of ITC Specialist Seminar on Internet Traffic Engineering and Traffic Management, Wurzburg, Germany, 2003.

[25] S. Joel, *et al.*, "Improving accuracy in end-to-end packet loss measurement," presented at the Proceedings of the 2005 conference on Applications, technologies, architectures, and protocols for computer communications, Philadelphia, Pennsylvania, USA, 2005.

[26] P. Vern, "Strategies for sound internet measurement," presented at the Proceedings of the 4th ACM SIGCOMM conference on Internet measurement, Taormina, Sicily, Italy, 2004.

[27] M. T. Y. Kitatsujji, S. Katsuno, and Y. Oie, "Usefulness of precise time-stamping for exposing network characteristics on high-speed links," presented at the In Proc. of SPIE ITcom, Philadelphia, USA, 2004.

[28] A. T. D. Mills, and B.C. Huffman, "Internet timekeeping around the globe," presented at the In Proc. Precision Time and Time Interval (PTTI) Applications and Planning Meeting, Long

Beach, CA, 1997.

[29] P. Attila, *et al.*, "PC based precision timing without GPS," presented at the Proceedings of the 2002 ACM SIGMETRICS international conference on Measurement and modeling of computer systems, Marina Del Rey, California, 2002.

[30] M. H. R. Hoeller, G. Griedling, N. Keroe, U. Schmid, K. Schossmaier, "SynUTC – High Precision Time Synchronization over Ethernet Networks," presented at the In Proceedings of the 8th Workshop on Electronics for LHC Experiments, Colmar, France, 2002.

[31] J. Micheel, *et al.*, "Precision timestamping of network packets," *Imw 2001: Proceedings of the First Acm Sigcomm Internet Measurement Workshop*, pp. 273-277 311, 2001.

[32] V. Srinivasan, *et al.*, "Fast and scalable layer four switching," presented at the Proceedings of the ACM SIGCOMM '98 conference on Applications, technologies, architectures, and protocols for computer communication, Vancouver, British Columbia, Canada, 1998.

[33] A. Feldman and S. Muthukrishnan, "Tradeoffs for packet classification," in *INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, 2000, pp. 1193-1202 vol.3.

[34] M. B. Milind, *et al.*, "Space Decomposition Techniques for Fast Layer-4 Switching," presented at the Proceedings of the IFIP TC6 WG6.1 \& WG6.4 / IEEE ComSoc TC on on Gigabit Networking Sixth International Workshop on Protocols for High Speed Networks VI, 2000.

[35] F. Baboescu, *et al.*, "Packet classification for core routers: is there an alternative to CAMs?," in *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications Societies. IEEE*, 2003, pp. 53-63 vol.1.

[36] P. Gupta and N. McKeown, "Packet classification using hierarchical intelligent cuttings," *IEEE Micro*, vol. 20, p. 34~41, 2000.

[37] S. Sumeet, *et al.*, "Packet classification using multidimensional cutting," presented at the Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications, Karlsruhe, Germany, 2003.

[38] T. V. Lakshman and D. Stiliadis, "High-speed policy-based packet forwarding using efficient multi-dimensional range matching," presented at the Proceedings of the ACM SIGCOMM '98 conference on Applications, technologies, architectures, and protocols for computer communication, Vancouver, British Columbia, Canada, 1998.

[39] B. Florin and V. George, "Scalable packet classification," vol. 13, ed: IEEE Press, 2005, pp. 2-14.

[40] V. Srinivasan, *et al.*, "Packet classification using tuple space search," presented at the Proceedings of the conference on Applications, technologies, architectures, and protocols for computer communication, Cambridge, Massachusetts, United States, 1999.

[41] S. Ed, *et al.*, "Packet Classification Using Extended TCAMs," presented at the Proceedings of the 11th IEEE International Conference on Network Protocols, 2003.

[42] V. C. Ravikumar, "EaseCAM: An Energy and Storage Efficient TCAM-Based Router Architecture for IP Lookup," vol. 54, ed: IEEE Computer Society, 2005, pp. 521-533.

[43] F. Zane, *et al.*, "Coolcams: power-efficient TCAMs for forwarding engines," in *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications Societies. IEEE*, 2003, pp. 42-52 vol.1.

[44] 任丰原，林闯，刘卫东, "IP 网络中的拥塞控制," *计算机学报*, vol. 26, 2003.

[45] V. Jacobson, "Congestion avoidance and control," *SIGCOMM Comput. Commun. Rev.*, vol. 25, pp. 157-187, 1995.

[46] W. Stevens, "TCP slow start, congestion avoidance, fast retransmit, and fast recovery algorithms," RFC 2001, January 19971997.

[47] S. Floyd and T. Henderson, "RFC2582: The NewReno Modification to TCP's Fast Recovery Algorithm," *RFC Editor United States*, 1999.

[48] M. Mathis, *et al.*, "RFC2018: TCP Selective Acknowledgement Options," *RFC Editor United States*, 1996.

[49] L. Brakmo, "TCP Vegas: End to end congestion avoidance on a global Internet," *IEEE Journal on selected Areas in communications*, vol. 13, p. 1465, 1995.

[50] S. Floyd, *et al.*, "Adaptive RED: An algorithm for increasing the robustness of RED's active queue management," *Preprint, available at http://www. icir. org/floyd/papers. html*, 2001.

[51] F. M. Anjum and L. Tassiulas, "Balanced-RED: An algorithm to achieve fairness in the Internet," in *Proceedings of IEEE INFOCOM1999*, New York, USA, 1999.

[52] A. K. Parekh and R. G. Gallager, "A Generalized Processor Sharing Approach to Flow-Control in Integrated Services Networks - the Single Node Case," *Ieee Infocom 92 - the Conference on Computer Communications, Proceedings Vols 1-3*, pp. 915-924 2515, 1992.

[53] H. Shimonishi and H. Suzuki, "Performance analysis of Weighted Round Robin cell scheduling and its improvement in ATM networks," *Ieice Transactions on Communications*, vol. E81b, pp. 910-918, May 1998.

[54] A. Demers, *et al.*, "Analysis and simulation of a fair queueing algorithm," *ACM SIGCOMM Computer Communication Review*, vol. 19, pp. 1-12, 1989.

作者简介：

罗腊咏： 中国科学院计算技术研究所，网络技术研究中心，博士研究生 luolayong@ict.ac.cn

谢应科： 中国科学院计算技术研究所，网络技术研究中心，高级工程师，硕士生导师

谢高岗： 中国科学院计算技术研究所，网络技术研究中心主任，研究员，博士生导师